APPLICATION FOR A UNITED STATES PATENT

For

VARIOUS METHODS AND APPARATUSES FOR A COMMAND BASED BUILT IN SELF TEST FOR MEMORIES

Inventors:

Yervant Zorian, Gevorg Torjyan, and Karen Darbinyan

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
32400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(408) 720-8300

Attorney's Docket No.: 4640.P019

"Express Mail" mailing label number: EV336584845US

Date of Deposit: September 16, 2003

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, PO Box 1450, Alexandria, Virginia 22313-1450

Deborah A. McGovern
(Typed or printed name of person mailing paper or fee)

DAMcGovern
(Signature of person mailing paper or fee)

September 16, 2003
(Date sign d)

VARIOUS METHODS AND APPARATUSES FOR A COMMAND BASED BIST FOR MEMORIES

NOTICE OF COPYRIGHT

FIELD OF THE INVENTION

[002] Embodiments of the invention generally relate to self testing of memories. More particularly, an aspect of an embodiment of the invention relates to self testing of memories embedded on a chip.

BACKGROUND OF THE INVENTION

[003] A built-in self-test (BIST) system of memories typically consists of a processor and a number of memories to be tested to find defects. In general during the BIST, the processor applies a set of test vectors to the memory cores. The processor then receives and evaluates the response to the test vectors to determine whether a defect exists in one or more word lines being tested in the memory. The processor may compare the results from the memory output with the predicted values to determine whether a defect exists.

[004] Figure 1 illustrates a processor sending built-in self-test (BIST) signals to multiple memories. The processor is configured to apply and execute a BIST

to various memories, such as memory 1, memory 2, and memory 3. The memories share the built-in self-test features and algorithms of the processor. The processor couples the self-test information via a parallel bus to each memory. A large number of routing paths exist between the processor and each memory, such as twenty-five routing lines in parallel with each memory. The processor connects data lines, address lines, control lines, and march algorithm lines in parallel with each memory.

[005] The information and routing paths pertaining to the self-test of one of the memories could be as follows. Several lines could carry test input data bits. Several lines could carry expected data bits back to the processor. Several lines in parallel to the data lines may carry the march algorithm steps. Several lines may carry the address information on which word line or word lines are to be tested. All of that information from the processor needs to be received by the memory in parallel so that the desired target memory can be vector tested to determine whether a defect exists or not in the memory cells in the memory.

[006] The processor and memories clock cycle may be synchronized such that on each clock cycle the processor provides the necessary data and control information to test an operation on a single address or all of the addresses in a memory under test. An intelligence wrapper surrounds each memory. The processor sends all of the information needed to conduct a march sequence of built-in self-test in parallel on a particular word line to each intelligence wrapper.

[007] This parallel architecture for sending BIST information works fine for a small number of memories that have a large byte capacity. Each memory takes

up a large amount of routing paths, such as 25, from the processor. The accumulation of all of the BIST routing paths for all of the memories takes up a lot of space on a system on a chip. However, since there may be only a few large memories on a system of a chip, the cost benefit analysis is fine. However, in a system on a chip design that utilizes a large amount of small memories, such as register files, this many routing paths per memory may not be acceptable.

[008] Another disadvantage of a parallel architecture from the processor to the memories may be that both operate in a synchronous timing cycle. The operations sent from the processor in the parallel lines occur at the clock speed of the memory. Each clock cycle, the processor sends all of the information in parallel needed to conduct a March sequence of the built-in self-test on the memory. Those operations then are executed on the memory at the speed of the memory. Those operations are also evaluated at the speed of the memory. Thus, the built-in self-test processor is generally built to run at the same clock speed as the memories during the built in self-testing. Building a built-in self-test processor that operates at a lower frequency than the memories may be significantly easier.

SUMMARY OF THE INVENTION

[009] Methods and apparatuses in which two or more memories share a processor for Built In Self Test algorithms and features are described. The processor initiates a Built In Self Test for the memories. Each memory has an intelligence wrapper bounding that memory. Each intelligence wrapper contains control logic to decode a command from the processor. Each intelligence wrapper contains logic to execute a set of test vectors on a bounded memory. The processor sends a command based self-test to each intelligence wrapper at a first clock speed and the control logic executes the operations associated with that command at a second clock speed asynchronous with the first speed. The processor loads the command containing representations of a march element and data to one or more of the intelligence wrappers via a serial bus.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The drawings refer to embodiments of the invention in which:

figure 1 illustrates a processor sending built-in self-test (BIST) signals to multiple memories;

figure 2 illustrates an embodiment of a processor issuing command based BIST instructions to two or more memories via a main serial bus;

figure 3 illustrates an embodiment of a self-test and repair processor having BIST logic encoded in the processor connecting via a serial control bus to multiple memories;

figure 4 illustrates an embodiment of a BIST processor connecting via a serial interface to the logic in the intelligence wrapper;

figures 5a through 5d illustrate embodiments of a structure of the command; and

figure 6 illustrates an example process of generating an embedded memory from designs of memory components with an embodiment of a memory compiler.

[0011] While the invention is subject to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and will herein be described in detail. The invention should be understood to not be limited to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention.

DETAILED DISCUSSION

[0012] In the following description, numerous specific details are set forth, such as examples of specific data signals, named components, connections, number of memories, etc., in order to provide a thorough understanding of the present invention. It will be apparent, however, to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well known components or methods have not been described in detail but rather in a block diagram in order to avoid unnecessarily obscuring the present invention. Further, specific numeric references such as first clock speed, may be made. However, the specific numeric reference should not be interpreted as a literal sequential order but rather interpreted that the first clock speed is different than a second clock speed. Thus, the specific details set forth are merely exemplary. The specific details may be varied from and still be contemplated to be within the spirit and scope of the present invention.

[0013] In general, methods and apparatuses in which two or more memories share a processor for Built In Self Test algorithms and features are described. The processor initiates BIST for the memories. Each memory has an intelligence wrapper bounding that memory. Each intelligence wrapper contains control logic to decode a command from the processor. Each intelligence wrapper contains logic to execute a set of test vectors on a bounded memory. The processor sends a command based self-test to each intelligence wrapper at a first clock speed. The control logic in the intelligence wrapper executes the operations associated with that command at a second clock speed

asynchronous with the first speed. The processor loads the command containing representations of one or more march elements, data, and other similar information to one or more of the intelligence wrappers via a serial bus. A march algorithm may be composed of a sequence of one or more march elements. In an embodiment, one march element exists per command.

[0014] Figure 2 illustrates an embodiment of a processor issuing command based BIST instructions to two or more memories via a main serial bus. The processor 202 connects via the main serial bus 204 and with a few parallel control lines 206 with the two or more memories, such as the first memory 208 to the Nth memory 224. Each memory has an intelligence wrapper bounding that memory such as the first intelligence wrapper 226 to the Nth intelligence wrapper 242. The built-in self-test processor 202 initiates a built-in self-test for each of the memories 208-224. The processor 202 loads a command containing representations of one or more march elements, and input data, expected results data, address information, and/or other similar information to each intelligence wrapper 226-242 via the serial bus 204. Note, all of the memories may have control lines, but the shown control lines 206 are limited to the third memory 212 and the eighth memory 222 for ease of understanding. The serial bus 204 and control lines 206 may also be called processor-wrapper nets.

[0015] All of the memories 208-224 may share the processor 202 for the built-in self-test features and algorithms. The intelligence wrapper for each memory contains control logic to decode the command from the processor. The intelligence wrapper for each memory contains control logic on how to execute a

set of test vectors on the bounded target memory to detect defects in that memory. For example, the first intelligence wrapper 226 contains a first logic 244. The processor 202 loads the command containing representations of the march element, the input data, the expected data, and address information specifying where to apply data to addresses in the memory via the serial interface. The command contains representations of the march element(s), test data, etc. coded in a compressed format. The representations may be the same information but in a compressed format. The control logic expands the representations of the march element(s), the input data, the expected output data, and address information on where to apply data to addresses in the memory to its full-uncompressed form.

[0016] The processor may have a few lines routing between the processor 202 and each memory 208-224 and intelligence wrapper 226-242. The routing paths may be the main serial bus 204 with a few parallel routing paths for additional control lines 206. For example, six routing paths for self test purposes may exist between the processor and each bounded memory to be tested. Since all of the data and instructions required to execute a march algorithm for a built-in self-test may load serially via one main serial bus, the number of required routing paths between the processor and each memory can be drastically reduced.

[0017] Further, since the processor 202 sends the information via a command serially, the command may be sent to the intelligence wrapper at a first clocking speed, such as 10-30 mega hertz. The control logic in the

intelligence wrapper receives the command, decodes the command, and executes operations associated with that command at a second clocking speed such as 600 or 700 mega hertz. The first speed is asynchronous with the second speed meaning that the processes occur at different speeds.

[0018] The control logic may be implemented as a state machine configured to decode and execute the set of test vectors coming from the processor 202. The processor 202 sends the commands in a compressed form and the state machine expands the representations of information in the command to their full-uncompressed form. In some prior art, the processor would send all the vector test data in an expanded form across the parallel lines to test the memory.

[0019] In the command, a representation of input or expected data may be a cipher, such as a 1 and 0 or a 1 and an A, which can then be looked up in a data table to determine the uncompressed bits and operations that correspond to the compressed information. When the control logic decodes the data representation for that memory, then the data representation may expand out from the two bit cipher to be a sequence of 16 bits of 1's and 0's or 32 bits of 1's and 0's depending upon the width of the word line being tested. Thus, the command may contain input data representations on the data strings to be applied to the memory being tested. The command may also contain expected output data from the memories.

[0020] Similarly, the command may include a compressed form, such as ciphers, of march algorithm sequences, address information on the memories to be tested in a compressed form, etc., which can then be decompressed and

expanded so that these operations may be carried out by logic within the intelligence wrapper.

[0021] March algorithms may be memory operations such as a read operation or a write operation. March algorithms may also indicate whether a single word line address is being tested or all of the word lines in the entire memory are being tested. March algorithms may also indicate whether the entire memory is tested with these read/write operations from the highest address to the lowest address or the lowest address to the highest address. Thus, the command may have representations of the read/write operations to be performed and the sequence of those operations to be performed on the memory. Further, the command may also include address information specifying which memories or word lines are to be tested. The command may also contain other similar information.

[0022] The processor 202 serially sends a command to one or more of the memories and the intelligence wrappers surrounding each memory. Logic in the intelligence wrapper may begin to decode each bit in the command as the logic receives each bit. Once the entire command has been received at the logic, then the processor may send an instruction to the intelligence wrapper to execute that command.

[0023] Figure 3 illustrates an embodiment of a self-test and repair processor having BIST logic encoded in the processor connecting via a serial control bus to multiple memories. The processor and each intelligence wrapper may enable a complete embedded memory self-test and repair function to be included on-chip.

The Self-Test-And-Repair (STAR) processor 302 contains logic 344 to determine which self test to apply and sends a command containing a compressed form of that information via a serial interface 304 to each of the intelligence wrappers 326-342. Each memory 308-324 having an intelligence wrapper 326-342 surrounding that memory. Each intelligence wrapper containing logic on how to execute the self test in the command.

[0024] The built-in self-test of the memory detects any defects in the memory core and may be conducted asynchronous to the operation of the processor. Redundancy allocation logic 346 contained in the processor 302 may have an algorithm to allocate the one or more redundant components in each memory, such as redundant columns and redundant rows, while fault testing each memory. The repair fuse box 348 stores the repair signature for each memory.

[0025] The intelligence wrapper that bounds each memory may contain logic configured specific to the particulars of that memory as well as logic that needs to be conducted at a high clock speed. For example, the size of each memory may be different, such as 512 kilobytes or 256 kilobytes, and each intelligence wrapper may contain its own address counter. Similarly, logic on how to execute the march algorithm on that particular memory may be in the intelligence wrapper. Logic that is common to all of the memories during a self-test may be stored in the shared processor such as logic to determine the type of march algorithm to apply to the memory under test, the shared March algorithm container, and other similar shared logic.

[0026] Figure 4 illustrates an embodiment of a BIST processor connecting via a serial interface to the logic in the intelligence wrapper. Several routing paths 406 exist between the processor 402 and each intelligence wrapper 426, 428. A first routing path 404 may be a command line, such as a scan-in command line, that serially connects the processor 402 to the command scan chain logic 450 to facilitate communicating the command from the processor 402 to the scan chain register 450 in the intelligence wrapper 426. The second routing path 452, such as a command scan en enable line, communicates to that scan chain register 450 when to start receiving bits of the command signal from the processor 402. The third parallel routing path 454 from the processor 402, such as a command force line, communicates to the control logic 444, such as an execution state machine, inside the intelligence wrapper 426 when to expand the built-in self-test information from the command. The logic in the intelligence wrapper may also start decoding bits from the command as soon as those bits are stored in the scan chain register 450. The command force line may also be used to send the execution signal of the command to execute the BIST test on the memory core 408 once all of the bits in the entire command have been sent serially to the scan chain register 450.

[0027] The processor 402 may contain logic 456 on determining which algorithms to run for the built-in self-test such the read/write sequences etc., the data vectors, address counting logic, etc. The address counter logic supplies and determines the address in the memory for the next set of tests. The processor 402 may contain logic 456 to compress information used in a self test

of the memories embedded on the chip. The processor 402 may contain logic 456 to communicate the compressed information in a serial manner to logic bounding each of the memories.

[0028] Logic on how to execute the built-in self-test algorithms may be decoupled from the processor and placed into the intelligence wrapper. The state machine 444 in the intelligence wrapper 426 may be configured on how to carry out the execution of the BIST algorithms received from the processor 402 for that specific memory 408. As discussed, the intelligence wrapper 426 is located very close to the memory core 408 and bound around the memory core 408. Thus, the logic in the intelligence wrapper 426 may operate at a high speed, such as the clock speed of the memory 408, to perform the various BIST test on that memory. The logic in the intelligence wrapper may operate at a clock speed greater than the clock speed of the processor 402.

[0029] The processor 402, through the serial interface bus 406 loads the command containing the representations of the march algorithm, data, and address representations into the scan chain register 450 in a serial manner. If twenty bits exist in the command, then after twenty clock cycles the command will be fully loaded into the scan chain register 450. After the initial bit is loaded into the scan chain register 450, the logic within the intelligence wrapper 426 may begin decoding the loaded bits on every clock cycle. After all, for example, twenty bits, load, then the command will be fully loaded into the scan chain register 450. The processor 402 may direct the execution of that command by applying a command force signal through the command force routing path 454 to

the state machine 444 in the intelligence wrapper 426. The logic in the intelligence wrapper 426 executes the command from the processor 402, thereby executing the next sequence of the march algorithm on the memory 408.

[0030] The intelligence wrapper may contain address generation logic 458, data generation logic 460, control generation logic 462, data comparison logic 464, as well as the command execution state machine 444, the command scan chain register 450, and other similar logic. The address generation logic 458 may generate the X and Y coordinates of the memory word line to be tested. The data generation logic 460 may expand the representation of the data input in the command to generate the sequence of data to be tested for that particular memory 408 and the number of bits needed to test a word length in that memory 408. For example, 16-bit word length, 32-bit word length etc. The data generation logic 460 may receive a representation of the input data such as the cipher 1A. Based off of that 1A representation from the command, the data generation logic 460 may then generate a sequence of 16 1 and 0's to word line being tested. If the cipher were, for example, a 1 and a B, the data generation logic 460 would then generate a different sequence of 1's and 0's for that 16-bit word line being tested. The control generation logic 462 controls the read/write operations on the selected addresses in the memory 408 based upon the march algorithm contained within the command. The control generation logic 462 controls how those read and write operations are performed on the memory 408 and whether it is a specific memory word line being tested or all of the word lines in the memory 408 are being tested. The control generation logic 462 may also

control the direction of sequences of operations on the memory 408. The control generation logic 462 may determine whether the memory 408 is being tested from the highest memory address to the lowest memory address, or from the lowest memory address to the highest memory address based upon the march algorithm contained within the command.

[0031] Data comparison logic 464 is configured to compare the actual vectors at an output of the bounded target memory 408 with the predicted vector results provided by the command. The data comparison logic 464 may be, for example, an Exclusive Or gate (XOR) series knowing and comparing what the referenced values should to be to what the actual output test data from the memory is.

[0032] Sequences of march algorithms may be carried out to perform an entire built-in self-test to detect all of the defects in a memory 408. A march algorithm in the command may instruct the control generation logic 462 to conduct a single operation, such as a read operation, or may instruct the control generation logic 462 to perform several operations back to back such as a read operation, then a write operation, followed by another write operation on a given word line all back to back. Then every word line tested will then perform that series of operations, a read, write, and a write on each word line in that memory 408. The intelligence wrapper 426 does not need to wait to receive another serially fed command to perform those three operations in rapid succession.

[0033] As discussed, the processor 402 may operate asynchronously from the clock of the memory 408 and thus be able to run at a lower frequency than

the memory, which may be much easier to design. The commands from the processor may be executed by the logic in the intelligence wrapper 426 at the higher frequency memory speed and transferred serially from the processor 402 at the bus speed. Because the logic in the intelligence wrapper 426 may run at a higher speed than a prior parallel architecture coming across the bus, the BIST test run may take nearly the same time or even less time then some prior techniques.

[0034] The processor 402 may connect to greater amount of memories 426, 428 using a command based BIST architecture because a fewer number of routing lines 406 need to exist between each memory and the processor. The BIST resources on a system on a chip take up less space then some prior techniques because a greater amount of memories may be connected per processor and fewer routing paths are needed.

[0035] Figures 5a through 5d illustrate embodiments of a structure of the command. A command may be an accumulation of various blocks of interrelated information. Referring to figures 5a –5c, a structure of the command may have blocks of interrelated information such as a SA block 570 indicating whether the march algorithm must be executed on a single memory address, or through the entire memory space from the highest memory address to the lowest memory address in that memory or vice versa. The next block of information in that command could be an address direction block 572 indicating whether the address is going to be incrementing/counting up in memories or decrementing down in memories during the march algorithm execution. The next block of

information may be a number of actions (i.e. operations) block 574 such as a read operation, another read operation etc. This number of actions block 574 could indicate that a single operation is to occur or point to another block of information stored in the code of the logic in the intelligence wrapper, such as Element descriptor block 576 for execution of multiple back to back operations. The element descriptor block 576 may alternatively be transmitted via the command. Operations, for example, may be a read operation, a write operation, or a read/write operation or similar operation.

[0036] Referring to Figure 5b, the element descriptor block 576 may be tied a number of actions to be performed on the memory. For example, the first action descriptor block 578 may list an initial operation to be performed on the memory such as a read operation, the second action descriptor block 580 may list the next operation to be performed on the memory such as a write operation. The Nth action descriptor block 582 may list the final back to back operation to be performed on the memory.

[0037] Each action descriptor block 578-582 may point to the data descriptor blocks 584, 586 of information shown in figure 5c. The data descriptor block 584 may provide the input data for that operation. The data descriptor block 586 may provide the expected output data for that operation based upon the input test vectors. Thus, an action description block may contain a field such as CD that contains the data pattern, which the logic uses during that particular read/write operation. The entire BIST test may be composed of thousands of commands.

[0038] Alternatively, a structure of a command does not need to be a loose conglomeration of blocks of information related to each another. Referring to Figure 5d, the structure of the command may be a series of bits and the position of a bit within that string of sequential bits associates that bit with particular section of information. A command may be such that, for example, the initial N number of bits 588 in the command represent the addresses of the memory being tested and the direction of that march algorithm, the next N number of bits 590 represent the number and type of operations to be performed, the next N number of bits 592 represent in the command may correspond to the input data representation that will be used, the next sequential N number of bits 594 represent may be the expected data results, etc.

[0039] Figure 6 illustrates an example process of generating an embedded memory from designs of memory components with an embodiment of a memory compiler.

[0040] In block 605, the designs for each processor and memory component for the embedded memory are supplied to the memory compiler, as well as other design parameters such as the number of columns and number of rows of the array, the available size on the chip and other parameters. Thus, the designs for one or more memories that share a processor for Built In Self Test may be supplied to the memory compiler. A memory compiler may be a software program comprised of multiple algorithms and designs for the purpose of generating a circuit design and a layout in a space available on a target chip. The set of application-specific algorithms and interfaces of the memory compiler

may be used by system IC integrators to rapidly create hundreds of silicon-proven memory cores. The memory compiler receives the memory component designs and utilizes those memory component designs in conjunction with memory circuit designs to optimize a circuit design and layout in the space available on a target chip.

[0041] In block 610, the memory compiler generates a netlist and a layout targeted to fit in the space available on a target chip. The memory compiler stores the data representing the embedded memory typically on a machine-readable medium. The memory compiler selects the memory component building blocks so that they are sized appropriate for the targeted fabrication technology. The memory compiler then provides the memory layout to be used to generate one or more lithographic masks to be used in the fabrication of that embedded memory. The memory compiler also provides a netlist for verification of the embedded memory.

[0042] In block 615, the memory layout generated is integrated with the rest of the layout for the chip and a machine generates the lithographic masks that contain the information necessary for the fabrication of a functional device. The machine generates one or more lithographic masks to be used to transfer that circuit design onto the chip. The memory solution for embedded applications integrates easily with the standard single poly CMOS processes.

[0043] In block 620, a fabrication facility fabricates the chips with the embedded memories using the lithographic masks generated from the memory compiler's circuit design and layout. Fabrication facilities may use a standard

CMOS logic process having minimum line widths such as 1.0 um, 0.50 um, 0.35 um, 0.25 um, 0.18 um, 0.13 um, 0.10 um, 90 nm, or less, to fabricate the chips. The size of the CMOS logic process employed typically defines the smallest minimum lithographic dimension that can be fabricated on the chip using the lithographic masks, which in turn determines minimum component size. In an embodiment, light is shown through these lithographic masks onto the chip to transfer the circuit design and layout for the embedded memory onto the chip itself. In an embodiment, the memory compiler is designed for embedded applications in the standard CMOS logic process.

[0044] In one embodiment, the software used to facilitate the memory compiler can be embodied onto a machine-readable medium. A machine-readable medium includes any mechanism that provides (e.g., stores and/or transmits) information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; DVD's, electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, EPROMs, EEPROMs, FLASH, magnetic or optical cards, or any type of media suitable for storing electronic instructions. Slower mediums could be cached to a faster, more practical, medium.

[0045] In an embodiment, an example memory compiler may comprise the following. A graphic user interface, a common set of processing elements, and a library of files containing design elements such as circuits, control logic, and cell

arrays that define the complier. In an embodiment, object code in a set of executable software programs.

[0046] As noted, in an embodiment, a designer chooses the specifics of the memory configuration to produce a set of files defining the requested memory instances. A memory instance may include front end views and back end files. The front end views support documentation, simulation, debugging, and testing. The back end files, such as a layout, physical LEF, etc are for layout and fabrication.

[0047] The memory complier outputs may include Behavioral Models and Test Benches (Verilog, VHDL), •Timing Models (TLF, .Lib and STAMP), Test Models (MemBIST, FastScan), Structural Netlists (EDIF, Spice), Power Models (WattWatcher, ALF), Floorplanning and Place&Route Models, Physical LEF, FRAM, Layout (GDS), Datasheets (including power, timing, and area specifications, as well as other outputs. When programming occurs or if a revision is needed, the designer merely has to redesign the block, a few metal and via masks.

[0048] Some portions of the detailed descriptions above are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical

manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0049] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the above discussions, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers, or other such information storage, transmission or display devices.